

**Московский авиационный институт
(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

Лабораторная работа № 3

Тема: Механизмы наследования в C++

Студент: Пивницкий Даниэль
Сергеевич

Группа: 80-206

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи

Разработать классы согласно варианту задания, классы должны наследоваться от базового класса Figure. Фигуры являются фигурами вращения. Все классы должны поддерживать набор общих методов:

1. Вычисление центра фигуры;
2. Вывод в стандартный поток вывода `std::cout` координат вершин фигуры;
3. Вычисление площади фигуры;

Создать программу, которая позволяет:

- Вводить из стандартного ввода `std::cin` фигуры, согласно варианту задания. (6-угольник, 8-угольник, Треугольник)
- Сохранять созданные фигуры в динамический массив `std::vector<Figure*>`
- Вызывать для всего массива общие функции (1-3 см. выше). Т.е. распечатывать для каждой фигуры в массиве геометрический центр, координаты вершин и площадь.
- Необходимо уметь вычислять общую площадь фигур в массиве.
- Удалять из массива фигуру по индексу;

2. Описание программы

Изначально создается базовый класс Figure задающий общий принцип структуры для классов — наследников(треугольника, 6-угольника и 8-угольника).

Наследование позволяет избежать дублирования лишнего кода при написании классов, т. к. класс может использовать переменные и методы другого класса как свои собственные.

В данном случае класс Figure является абстрактным — он определяет интерфейс для переопределения методов другими классами.

3. Набор тестов

test_1

add

1

1 1

0 0

1 0

add

2

2 0

0 1

1 2

2 3

3 4

4 5

add

3

0 0

1 1

2 2

3 3

4 4

5 5

6 6

7 7

out

quit

test_2

add

1

0 0

2 2

0 1

add

2

0 0

1 1

2 2

3 3

4 4

5 5

add

3

0 1

1 2

2 3

3 4

4 5

5 6

6 7

7 8

out

quit

4. Результаты выполнения тестов

test_1

Operations: add / delete / out / quit

add

Fig types: 1 - triangle; 2 - hexagon; 3 - octagon

1

0 0

2 2

0 1

add

Fig types: 1 - triangle; 2 - hexagon; 3 - octagon

2

0 0

1 1

2 2

3 3

4 4

5 5

add

Fig types: 1 - triangle; 2 - hexagon; 3 - octagon

3

0 1

1 2

2 3

3 4

4 5

5 6

6 7

7 8

out

0:

Area: 1

Center: (0.666667;1)

Coordinates: (0;0) (2;2) (0;1)

1:

Area: 0

Center: (2.5;2.5)

Coordinates: (0;0) (1;1) (2;2) (3;3) (4;4) (5;5)

2:

Area: 0

Center: (3.5;4.5)

Coordinates: (0;1) (1;2) (2;3) (3;4) (4;5) (5;6) (6;7) (7;8)

Total area: 1

test_2

Operations: add / delete / out / quit

add

Fig types: 1 - triangle; 2 - hexagon; 3 - octagon

1

0 0

2 2

0 1

add

Fig types: 1 - triangle; 2 - hexagon; 3 - octagon

2

0 0

1 1

2 2

3 3

4 4

5 5

add

Fig types: 1 - triangle; 2 - hexagon; 3 - octagon

3

0 1

1 2

2 3

3 4

4 5

5 6

6 7

7 8

out

0:

Area: 1

Center: (0.666667;1)

Coordinates: (0;0) (2;2) (0;1)

1:

Area: 0

Center: (2.5;2.5)

Coordinates: (0;0) (1;1) (2;2) (3;3) (4;4) (5;5)

2:

Area: 0

Center: (3.5;4.5)

Coordinates: (0;1) (1;2) (2;3) (3;4) (4;5) (5;6) (6;7) (7;8)

Total area: 1

quit

C:\Users\Daniel\source\repos\lab3\Debug\lab3.exe (процесс 11504) завершает работу с кодом 0.

Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".

Чтобы закрыть это окно, нажмите любую клавишу...

5. Листинг программы

```
//
//  main.cpp
//  lab3
//  Variant 15
//  M8o-206B-19
//  Created by Daniel Pivnitskiy on 10.10.2020.
//  github.com/SLAST1
//  Copyright © 2020 Daniel Pivnitskiy. All rights reserved.
//
/*
Создать программу, которая позволяет:
Вводить из стандартного ввода std::cin фигуры, согласно варианту задания.
(6-угольник, 8-угольник, Треугольник)
Сохранять созданные фигуры в динамический массив std::vector<Figure*>
Вызывать для всего массива общие функции (1-3 см. выше).
Т.е. распечатывать для каждой фигуры в массиве геометрический центр,
координаты вершин и площадь.
Необходимо уметь вычислять общую площадь фигур в массиве.
Удалять из массива фигуру по индексу;
*/

#include <iostream>
#include "figure.h"
#include "triangle.h"
```



```

#include "octagon.h"
#include "hexagon.h"
#include <vector>
#include <string>

void read_figt(std::vector<Figure*>& fig)
{
    int figt;
    Figure* f = nullptr;

    std::cout << "Fig types: 1 - triangle; 2 - hexagon; 3 - octagon\n";
    std::cin >> figt;
    if (figt == 1) {
        f = new Triangle(std::cin);
    }
    else if (figt == 2) {
        f = new Hexagon(std::cin);
    }
    else if (figt == 3) {
        f = new Octagon(std::cin);
    }
    else{
        std::cout << "Wrong. Try 1 - triangle, 2 - hexagon or 3 - octagon\n";
    }
    fig.push_back(dynamic_cast<Figure*>(f));
}

int main() {
    unsigned int index;
    double Tarea = 0;
    std::string operation;
    std::vector<Figure*> fig;

    std::cout << "Operations: add / delete / out / quit\n";

    while (std::cin >> operation) {
        if (operation == "add") {
            read_figt(fig);
        }
        else if (operation == "delete") {
            std::cin >> index;
            delete fig[index];
            for (; index < fig.size() - 1; ++index) {
                fig[index] = fig[index + 1];
            }
            fig.pop_back();
        }
        else if (operation == "out") {
            Tarea = 0;
            for (unsigned int i = 0; i < fig.size(); i++) {

```

```

        std::cout << i << ":\n";
        std::cout << "Area: " << fig[i]->area() << std::endl;
        std::cout << "Center: " << fig[i]->center() << std::endl;
        std::cout << "Coordinates: ";
        fig[i]->print(std::cout);
        std::cout << std::endl;
        Tarea += fig[i]->area();
    }
    std::cout << "Total area: " << Tarea << std::endl;
}
else if (operation == "quit") {
    for (unsigned int i = 0; i < fig.size(); ++i) {
        delete fig[i];
    }
    return 0;
}
else {
    std::cout << "Wrong. Operations: add / delete / out / quit\n";
}
}
}

```

```

//
// figure.h
// lab3
// Variant 15
// M8o-206B-19
// Created by Daniel Pivnitskiy on 10.10.2020.
// github.com/SLAST1
// Copyright © 2020 Daniel Pivnitskiy. All rights reserved.
//
#ifndef FIGURE_H
#define FIGURE_H

#include <iostream>
#include "point.h"

class Figure {
public:
    virtual double area() const = 0;
    virtual Point center() const = 0;
    virtual std::ostream& print(std::ostream& out) const = 0;
    virtual ~Figure() = default;
};

#endif

```

```

//
//  hexagon.cpp
//  lab3
//  Variant 15
//  M8o-206B-19
//  Created by Daniel Pivnitskiy on 10.10.2020.
//  github.com/SLAST1
//  Copyright © 2020 Daniel Pivnitskiy. All rights reserved.
//
#include "hexagon.h"
#include <cmath>

Hexagon::Hexagon() : A{ 0, 0 }, B{ 0, 0 }, C{ 0, 0 }, D{ 0, 0 }, E{ 0, 0 },
F{ 0, 0 } {}

Hexagon::Hexagon(std::istream& in) {
    in >> A >> B >> C >> D >> E >> F;
}

double Hexagon::area() const {
    return 0.5 * abs(A.X() * B.Y() + B.X() * C.Y() + C.X() * D.Y() + D.X()
* E.Y() + E.X() * F.Y() + F.X() * A.Y()
- B.X() * A.Y() - C.X() * B.Y() - D.X() * C.Y() - E.X() * D.Y()
- F.X() * E.Y() - A.X() * F.Y());
}

Point Hexagon::center() const {
    return Point{ (A.X() + B.X() + C.X() + D.X() + E.X() + F.X()) / 6,
(A.Y() + B.Y() + C.Y() + D.Y() + E.Y() + F.Y()) / 6 };
}

std::ostream& Hexagon::print(std::ostream& out) const {
    out << A << " " << B << " " << C << " " << D << " " << E << " " << F;
    return out;
}

//
//  hexagon.h
//  lab3
//  Variant 15
//  M8o-206B-19
//  Created by Daniel Pivnitskiy on 10.10.2020.
//  github.com/SLAST1
//  Copyright © 2020 Daniel Pivnitskiy. All rights reserved.
//
#ifndef HEXAGON_H
#define HEXAGON_H

```

```

#include "figure.h"
#include "point.h"

class Hexagon : public Figure {
public:
    Hexagon();
    Hexagon(std::istream& in);
    double area() const override;
    Point center() const override;
    std::ostream& print(std::ostream& out) const override;
private:
    Point A;
    Point B;
    Point C;
    Point D;
    Point E;
    Point F;
};

#endif

//
//  octagon.cpp
//  lab3
//  Variant 15
//  M8o-206B-19
//  Created by Daniel Pivnitskiy on 10.10.2020.
//  github.com/SLAST1
//  Copyright © 2020 Daniel Pivnitskiy. All rights reserved.
//
#include "octagon.h"
#include <cmath>

Octagon::Octagon() : A{ 0, 0 }, B{ 0, 0 }, C{ 0, 0 }, D{ 0, 0 }, E{ 0, 0 },
F{ 0, 0 }, G{ 0, 0 }, H{ 0, 0 } {}

Octagon::Octagon(std::istream& in) {
    in >> A >> B >> C >> D >> E >> F >> G >> H;
}

double Octagon::area() const {
    return 0.5* abs(A.X() * B.Y() + B.X() * C.Y() + C.X() * D.Y() + D.X()
* E.Y() + E.X() * F.Y() + F.X() * G.Y() + G.X() * H.Y() + H.X() * A.Y()
- B.X() * A.Y() - C.X() * B.Y() - D.X() * C.Y() - E.X() * D.Y()
- F.X() * E.Y() - G.X() * F.Y() - H.X() * G.Y() - A.X() * H.Y());
}

Point Octagon::center() const {
    return Point{ (A.X() + B.X() + C.X() + D.X() + E.X() + F.X() + G.X() +

```

```

H.X()) / 8, (A.Y() + B.Y() + C.Y() + D.Y() + E.Y() + F.Y() + G.Y() + H.Y()) /
8 };
}

```

```

std::ostream& Octagon::print(std::ostream& out) const {
    out << A << " " << B << " " << C << " " << D << " " << E << " " << F
<< " " << G << " " << H;
    return out;
}

```

```

//
// octagon.h
// lab3
// Variant 15
// M8o-206B-19
// Created by Daniel Pivnitskiy on 10.10.2020.
// github.com/SLAST1
// Copyright © 2020 Daniel Pivnitskiy. All rights reserved.
//
#ifndef OCTAGON_H
#define OCTAGON_H

```

```

#include "figure.h"
#include "point.h"

```

```

class Octagon : public Figure {
public:
    Octagon();
    Octagon(std::istream& in);
    double area() const override;
    Point center() const override;
    std::ostream& print(std::ostream& out) const override;
private:
    Point A;
    Point B;
    Point C;
    Point D;
    Point E;
    Point F;
    Point G;
    Point H;
};
#endif

```

```

//
// triangle.cpp
// lab3

```

```

// Variant 15
// M8o-206B-19
// Created by Daniel Pivnitskiy on 10.10.2020.
// github.com/SLAST1
// Copyright © 2020 Daniel Pivnitskiy. All rights reserved.
//
#include "triangle.h"
#include <cmath>

Triangle::Triangle() : A{ 0, 0 }, B{ 0, 0 }, C{ 0, 0 } {}

Triangle::Triangle(std::istream& in) {
    in >> A >> B >> C;
}

double Triangle::area() const {
    return 0.5 * abs(A.X() * B.Y() + B.X() * C.Y() + C.X() * A.Y() - B.X()
* A.Y() - C.X() * B.Y() - A.X() * C.Y());
}

Point Triangle::center() const
{
    return Point{ (A.X() + B.X() + C.X()) / 3, (A.Y() + B.Y() + C.Y()) / 3
};
}

std::ostream& Triangle::print(std::ostream& out) const
{
    out << A << " " << B << " " << C;
    return out;
}

//
// triangle.h
// lab3
// Variant 15
// M8o-206B-19
// Created by Daniel Pivnitskiy on 10.10.2020.
// github.com/SLAST1
// Copyright © 2020 Daniel Pivnitskiy. All rights reserved.
//
#ifndef TRIANGLE_H
#define TRIANGLE_H

#include "figure.h"
#include "point.h"

class Triangle : public Figure {

```

```

public:
    Triangle();
    Triangle(std::istream& in);
    double area() const override;
    Point center() const override;
    std::ostream& print(std::ostream& out) const override;
private:
    Point A;
    Point B;
    Point C;
};
#endif

```

```

//
// point.h
// lab3
// Variant 15
// M8o-206B-19
// Created by Daniel Pivnitskiy on 10.10.2020.
// github.com/SLAST1
// Copyright © 2020 Daniel Pivnitskiy. All rights reserved.
//
#ifndef POINT_H
#define POINT_H

#include <iostream>

class Point {
public:
    Point();
    Point(double x, double y);
    double X() const;
    double Y() const;
    friend std::ostream& operator<< (std::ostream& out, const Point& p);
    friend std::istream& operator>> (std::istream& in, Point& p);
private:
    double x;
    double y;
};
#endif

```

6. Выводы

В процессе выполнения лабораторной работы освоил основы полиморфизма и наследования в C++, опробовал различные модели наследования и применения в нем шаблонов, от которых позже отказался.

Также в данной лабораторной работе используется полиморфизм он осуществляется посредством виртуальных функций.

7. Список литературы

1.Руководство по написанию кода на С++ [Электронный ресурс].

URL:<https://metanit.com/cpp/tutorial/>

Дата обращения: 10.09.2019

2.Документация по С++ [Электронный ресурс]. URL:

<https://docs.microsoft.com/ru-ru/cpp>

Дата обращения 12.09.2019